Beeldverwerken Lab 4

Max Bos
10669027

Haischel Dabian
10336230

May 16, 2017

1 Introduction

Scale-invariant feature transform (SIFT) is an algorithm to detect and describe local features in images and in specific addresses the problem of matching features with changing scale and rotation. This report discusses the implementation of this algorithm to build a program that automatically generates a mosaic out of two images. To achieve this, SIFT is used to detect matching points between images and RANSAC is used to compute homographies between these images. Understanding of SIFT is obtained by carefully reading *Distinctive Image Features from Scale-Invariant Keypoints* (David G. Lowe, 2004).

2 Theory

2.1 SIFT

Scale-invariant feature transform (SIFT) is an algorithm to detect and describe local features in images, invariant to changes in image scale, rotation, noise and illumination. The descriptions of local features in images can be used to provide feature descriptions of characteristic points of an object depicted in an image. These descriptions can then be used to identify the same object in a test image, where the image could be subject to noise, and differing scale, roatation and illumination. The SIFT algorithm comprises four global stages, which are elaborated in the following subsections.

2.1.1 Scale-space extrema detection

This stage identifies all key locations in the image that are maxima and minima of the result of the Difference-of-Gaussian function applied in scale space to a series of smoothed and resampled images. These key locations attempt to identify feature points in the image, termed as keypoints. By obtaining the keypoints from a scale space, the keypoints become scale invariant. The scale space of an image is defined as a function, $L(x, y, k\sigma)$, that is produced from the convolution of a variable-scale Gaussian, $G(x, y, k\sigma)$ at scale $k\sigma$, with an input image, $I(x, y)$:

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y)$$

The convolved images are grouped by octave, where for each octave the value of σ is doubled. The value of k is selected so that a fixed number of convolved images per octave is obtained. The Difference-of-Gaussian images are taken from adjacent Gaussian-blurred images per octave.

To locate scale-space extrema, the derivative of the scale space function is used, which in the case of SIFT is calculated as the Difference-of-Gaussian (DoG) $D(x, y, \sigma)$, by subtracting two images with a slight difference in scale dimension (a difference of $k_i\sigma - k_j\sigma$):

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma)$$

The local maxima and minima of the difference-of-Gaussian images are detected by comparing the value of a pixel to its 26 neighbors in 3x3 regions at the current (8 neighbors) and adjacent scales (each 9 neighbors). If the pixel value is a minimum or maximum compared to its neighbors, the pixel is assumed to be an extrema.

This keypoint detection step is a variation of one of the blob detection methods developed by Lindeberg by detecting scale-space extrema of the scale normalized Laplacian. Instead the difference-of-Gaussian is used in SIFT, since it can be seen as an approximation to the Laplacian. The difference-of-Gaussian function is a relatively cheap operation compared to the Laplacian.

2.1.2 Keypoint localization

In the previous stage (Scale-Space Extrema Detection), many keypoints were eliminated by selecting the extrema of 3x3 regions. During the Keypoint Localization stage, points that have a low contrast, since these are sensitive to noise, and points that are poorly localized along an edge are eliminated. This elimination is achieved by performing a detailed fit to the nearby data for accurate location, scale, and ratio of principal curvatures.

First, for each candidate keypoint, an interpolated estimate for the location of the extremum is calculated. The function value at the extremum, is useful for rejecting unstable extrema with low contrast. This value is obtained is computed as the second-order Taylor expansion $D(\mathbf{x})$ at the location of the extremum $\hat{\mathbf{x}}$:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T \hat{\mathbf{x}}$$

In Lowe (2004) all extrema with a value of $D(\hat{\mathbf{x}}$ less than 0.03 were discarded.

Second, in order to increase stability, the keypoints with poorly determined locations and high edge response need to be eliminated. Foor poorly defined locations in the DoG function, the principal curvature across the edge would be larger than the principal curvature along it. The principal curvatures are found by solving the eigenvalues of the second-order Hessian matrix, H:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

The ratio of the two eigenvalues can be used to determine if a keypoint should be eliminated. Given a ratio threshold r_{th}, if the ratio for a candidate keypoint is larger than $(r_{th}+1)^2/r_{th}$, that keypoint is poorly localized and hence rejected.

2.1.3 Orientation assignment

This stage assigns one or more orientations based on local image gradient directions to each keypoint. Invariance to image rotation can be achieved by representing the keypoint descriptor relative to this orientation.

First, the Gaussian-smoothed image is used so the computations are performed in a scale-invariant manner. Then the gradient magnitude (m) and orientation (θ) are computed for each sample at that scale, using the following equations:

$$m(x,y) = \sqrt{L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/L(x+1,y) - L(x-1,y)))$$

Next a histogram with 36 bins is formed, with each bin covering 10 degrees, from the gradient orientations. The peaks in the histogram correspond to dominant orientations. The orientations that correspond to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the keypoint. Then a keypoint is constructed of the highest peak of this histogram.

2.1.4 Keypoint descriptor

The previous stages ensured invariance to image location, scale and rotation. This stage ensures that the keypoint is is highly distinctive and partially invariant to remaining variations, e.g. illumination and 3D viewpoint.

First, a set of orientation histograms is created on 4x4 pixel neighborhoods with 8 bins each. The descriptor is then formed from a vector containing all the values of the orientation histograms. The resulting element vector for each keypoint is therefore $4*4*8 = 128$ elements in size. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination. To reduce the effects of non-linear illumination a threshold of 0.2 is applied and the vector is again normalized.

2.2 RANSAC

Random sample consensus (RANSAC) is used to compute homographies between pairs of images. RANSAC is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers. RANSAC works under the assumption that the provided data contains inliers, data whose distribution can be explained by some set of model parameters.

The algorithm, according to Wikipedia, works as follows:

1. Select a random subset of the original data. Call this subset the hypothetical inliers.
2. A model is fitted to the set of hypothetical inliers.
3. All other data are then tested against the fitted model. Those points that fit the estimated model well, according to some model-specific loss function, are considered as part of the consensus set.
4. The estimated model is reasonably good if sufficiently many points have been classified as part of the consensus set.

5. Afterwards, the model may be improved by reestimating it using all members of the consensus set.

This procedure is repeated a fixed number of times.

3 Algorithm

3.1 SIFT

SIFT is implemented using `vlfeat` library. The SIFT features were calculated using `vl_sift` and matches between the two SIFT descriptor sets were found using `vl_ubcmatch`:

```
%% Find the matches using vl_feat
% Calculate the SIFT features in the images
[F1, D1] = vl_sift(imgNachtwacht1);
[F2, D2] = vl_sift(imgNachtwacht2);
% Match the two descriptor sets to find
% possible matching points in the two images.
matches = vl_ubcmatch(D1, D2);
```

3.2 RANSAC

The RANSAC algorithm is implemented as described on the RANSAC Wikipedia page. The previous Theory section contains the procedure of the algorithm. The code for the RANSAC implementation is globally structured as follows:

```
Given:
    iterationTimes { the maximum number of iterations allowed in the algorithm
    error { error margin
    threshold { threshold of number of inliers

Return:
    matrix { model parameters which best fit the data

coords1 = getCoordsImageOne
coords2 = getCoordsImageTwo
for i:iterationTimes
    points = selectRandomPoints(coords1)
    matrix = estimateMatrix(points)
    transFormedCoord = matrix * coords
    inliers = euclidiandistance(transFormedCoords, coords2) < error
    if inliers > threshold
        keepInliersAsDataset
return matrix
```

4 Experiments

Two experiments have been performed on the developed mosaicing functionality. The first experiment stiches `nachtwacht1.jpg` to `nachtwacht2.jpg` using SIFT and RANSAC. The

results of the experiment can be seen in Figure 1. The results of the second experiment where `ankor1.jpg` and `ankor2.jpg` were stitched can be seen in Figure 2. The implementation of the mosaic functionality seems to sometimes result in incorrect stitching, this could be due to the parameter settings. The experiments have been run using a maximum number of iterations of 20.

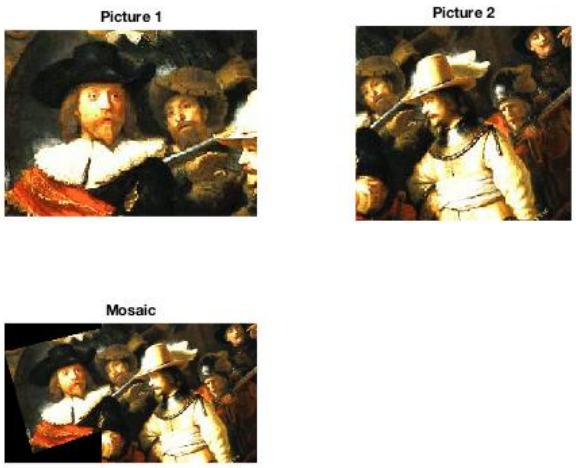

Figure 1: Mosaic of nachtwacht1.jpg and nachtwacht2.jpg

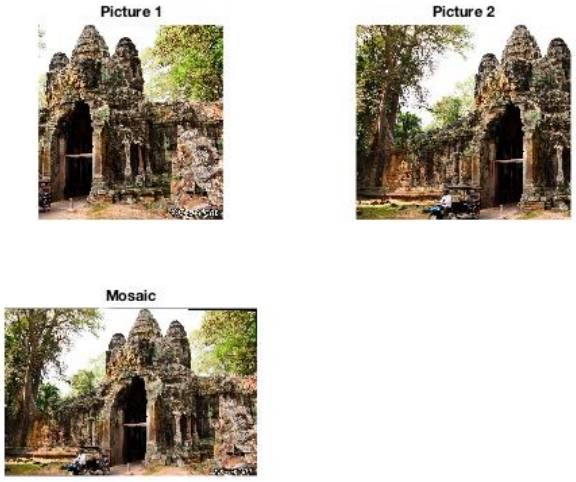

Figure 2: Mosaic of ankor1.jpg and ankor2.jpg

5 Conclusions

SIFT is a solid algorithm for indentifying matching local features (keypoints) in different images. The keypoint descriptors resulting from the SIFT algorithm can be effectively used for the application of image stitching in combination with RANSAC. The mosaic implementation achieved correct image stitching, although occasionaly the implementation would yield incorrect stitching. This could be due to suboptimal parameter settings.

Appendix

A Theory Answers

3. The term *scale space* theory refers to a framework which is used in computer vision. The scale space theory is used for handling image structures at different scales, by representing an image in a continuous space of smoothed images with differing smoothing kernel size (by parameterising by the size of the smoothing kernel). The space of differing smoothing kernel size allows visual operations to be made scale invariant, which is necessary for dealing with the size variations that may occur in image data. In Lowe's paper the scale space function was the Gaussian function, and the parameter which was changed to compute scales was σ.

   Scale space extrema refer to the maxima and minima value points at a certain scale space. These extrema are used to determine location and scale during keypoint localization in Lowe's SIFT algorithm. Intuitively, the scale space extrema are the most distinctive points in the image.

4. Lowe gives the following equation to efficiently detect stable keypoint locations in scale space:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

   According to Lowe, $\nabla^2 G$ can be computed from the finite difference approximation to $\frac{\partial G}{\partial \sigma}$:

$$\begin{aligned} \frac{\partial G}{\partial \sigma} &= \sigma \nabla^2 G \\ &\approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \end{aligned}$$

   Therefore:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$$

   To close the loop in the explanation, returning to D:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= (k - 1)\sigma^2 \nabla^2 G * I(x, y) \end{aligned}$$

5. To calculate the eigenvalues of a matrix A, one has to solve the following equation:

$$det(A - \lambda I) = 0$$

   Using the Hessian matrix as A gives:

$$(D_{xx} - \lambda)(D_{yy} - \lambda) - D_{xy}^2 = 0$$

Since f_{xy} is 0:

$$(D_{xx} - \lambda)(D_{yy} - \lambda) = 0$$
$$\lambda = D_{xx} \wedge \lambda = D_{yy}$$

Which proves that:

$$D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$

It is always possible to diagonalize the Hessian (H), since the Hessian is a square matrix of which exists an invertible matrix P such that $P^{-1}HP$ is a diagonal matrix.

6.
- The prior image smoothing for the first level of each octave (σ), may need to be varied for the mosaicing task. Since there is a cost to using a large σ in terms of efficiency, Lowe chooses $\sigma = 1.6$ which provides close to optimal repeatability. However, for the mosaicing task, a larger σ could improve performance.
- The number of scales sampled per octave, may needs to be varied for the mosaicing task. Lowe chooses to use 3 scale samples per octave, since the cost of computation rises with the number of scale samples per octave. According to Lowe, the success of object recognition often depends more on the quantity of correctly matched keypoints, as opposed to their percentage correct matching, for many applications it will be optimal to use a larger number of scale samples. Therefore, it is possible that a larger number of scale samples per octave is beneficial for the mosaicing task.

8. Lowe (2004) gives the following equation for the Taylor expansion (up to the quadratic terms) of the scale-space function, $D(x, y, \sigma)$, shifted so that the origin is at the sample point:

$$D(x) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{1}$$

According to Lowe, the location of the extremum, $\hat{\mathbf{x}}$, is determined by taking the derivative of this function with respect to x and setting it to zero, giving:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1}\frac{\partial D}{\partial \mathbf{x}} \tag{2}$$

Rewrite (1) and (2) to:

$$D(x) = D + a^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T B\mathbf{x} \tag{3}$$

and

$$\hat{\mathbf{x}} = -B^{-1}a \tag{4}$$

where $a = \frac{\partial D}{\partial \mathbf{x}}$ and $B = \frac{\partial^2 D}{\partial \mathbf{x}^2}$.

Substituting equation (4) into (3) gives

$$\begin{aligned} D(\hat{\mathbf{x}}) &= D + a^T(-B^{-1}a) + \frac{1}{2}(-B^{-1}a)^T B(-B^{-1}a) \\ &= D + a^T(-B^{-1}a) - \frac{1}{2}a^T(-B^{-1}a) \\ &= D + \frac{1}{2}a^T(-B^{-1}a) \\ &= D + \frac{1}{2}\frac{\partial D}{\partial \mathbf{x}}^T\hat{\mathbf{x}} \end{aligned}$$

This proves that the $\frac{1}{2}$ in the definition of the equation for $D(\hat{\mathbf{x}})$ on page 11 in Lowe's paper (2004) is not a typo.

9. We are in 3D because the descriptor consists of 3 parameters, namely, the (2D) x and y location coordinates and θ for the orientation. Thus, x and y denote the keypoint location and θ the keypoint orientation.

10. With affine changes in illumination is meant, the change in illumination on an image caused by an affine transformation (a linear transformation). Thus, the SIFT descriptor is invariant to **linear** illumination changes, since vector normalization is performed on the SIFT descriptor. In addition, brightness change in which a constant is added to each image pixel will not affect the gradient values, as they are computed from pixel differences.